Deep Learning Specialization
Structuring Machine Learning Projects

Du Ang
du2ang233@gmail.com

October 25, 2017

Contents

1 ML Strategy

1.1 Introduction to ML Strategy

1.1.1 Why ML Strategy?

90% is not good enough for your application, so you might try the following ideas to improve:

- Collect more data
- Collect more diverse training set
- Train algorithm longer with gradient descent
- Try Adam instead of gradient descent
- Try bigger network
- Tray small network
- Try dropout
- Add L_2 regularization
- Network architecture
  - Activation functions
  - # hidden units
  - ...

When trying to improve a deep learning system, you often have a lot of ideas or things you could try, ML strategy will offer you quick and effective ways to figure out which of all of these ideas are worth pursuing.

1.1.2 Orthogonalization

Orthogonalization or orthogonality is a system design property that assures that modifying an instruction or a component of an algorithm will not create or propagate side effects to other components of the system. It becomes easier to verify the algorithms independently from one another, it reduces testing and development time.

When a supervised learning system is design, these are the 4 assumptions that needs to be true and orthogonal.

1. Fit training set well in cost function
   - If it doesn't fit well, the use of a bigger neural network or switching to a better optimization algorithm might help.
2. Fit development set well on cost function
   - If it doesn't fit well, regularization or using bigger training set might help.
3. Fit test set well on cost function
   - If it doesn't fit well, the use of a bigger development set might help
4. Performs well in real world
   - If it doesn't perform well, the development test set is not set correctly or the cost function is not evaluating the right thing.

1.2 Setting up your goal

1.2.1 Single number evaluation metric

To choose a classifier, a well-defined development set and an evaluation metric speed up the iteration process.

Predict class $\hat{y}$	Actual class y: 1	Actual class y: 0
1	True positive	False positive
0	False negative	True negative

Figure 1: Example: Cat vs Non-cat. $y = 1$, cat image detected.

Precision Of all the images we predicted $y = 1$, what fraction of it have cats?

$$\text{Precision}(\%) = \frac{Turepositive}{Numberofpredictedpositive} \times 100 = \frac{Turepositive}{Truepositives + Falsepositives} \times 100$$

Recall Of all the images that actually have cats, what fraction of it did we correctly identifying have cats?

$$\text{Recall}(\%) = \frac{Truepositive}{Numberofpredictedactuallypositive} \times 100 = \frac{Turepositive}{Truepositive + Falsenegative} \times 100$$

The problem with using precision/recall as the evaluation metric is that you are not sure which one is better since in this case, both of them have a good precision et recall. F1-score, a harmonic mean, combine both precision and recall.

$$F_1 \text{ score} = \frac{2}{\frac{1}{p} + \frac{1}{r}}$$

1.2.2 Satisficing and Optimizing metric

There are different metrics to evaluate the performance of a classifier, they are called evaluation matrics. They can be categorized as satisficing and optimizing matrics. It is important to note that these evaluation matrics must be evaluated on a training set, a development set or on the test set.

Classifier	Accuracy	Running time
A	90%	80 ms
B	92%	95 ms
C	95%	1 500 ms

Figure 2: Example: Cat vs Non-cat, accuracy and running time are the evaluation metrics.

Accuracy is the optimizing metric, because you want the classifier to correctly detect a cat image as accuracy as possible. The running time which is set to be under 100ms in this example, is the satisficing metric which mean that the metric has to meet expectation set.

The general rule is

$$N_{\text{metric}} : \begin{cases} 1 & \text{Optimizing metric} \\ N_{\text{metric}} - 1 & \text{Satisficing metric} \end{cases}$$

1.2.3 Train/dev/test distributions

Setting up the training, development and test sets have a huge impact on productivity. It is important to choose the development and test sets from the same distribution and it must be taken randomly from all the data.

Guideline Choose a development set and test set to reflect data you expect to get in the future and consider important to do well.

1.2.4 Size of the dev and test sets

Old way of splitting data We had smaller dataset therefore we had to use a greater percentage of data to develop and test ideas and models.

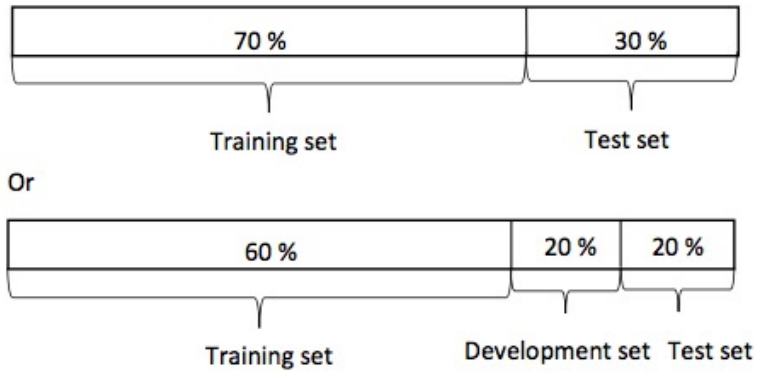

Figure 3: Old ways of splitting data

Modern era — Big data Now, because a large amount of data is available, we don't have to compromised as much and can use a greater portion to train the model.

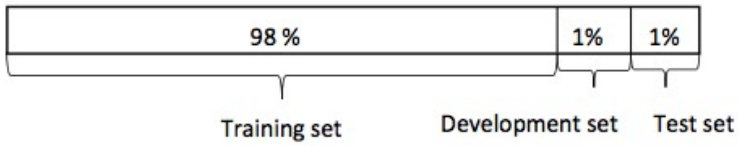

Figure 4: The way of splitting data in big data era

Guidelines

- Set up the size of test set to give a high confidence in the overall performance of the system.
- Test set helps evaluate the performance of the final classifier which could be less 30% of the whole dataset.
- The development set has to be big enough to evaluate different ideas.

1.2.5 When to change dev/test sets and metrics

Example: Cat vs Non-cat A cat classifier tries to find a great amount of cat images to show to to cat loving users. The evaluation metric used is a classification error, as it is shown in Figure 5.

Algorithm	Classification error [%]
A	3%
B	5%

Figure 5: The classification error of Algorithm A and Algorithm B

It seems that Algorithm A is better than Algorithm B since there is only a 3% error, however for some reason, Algorithm A is letting through a lot of pornographic images.

Algorithm B has 5% error thus it classifies fewer images but it doesn't have pornographic images. From a company's point of view, as well as from a user acceptance point of view, Algorithm B is actually a better algorithm. The evaluation metric fails to correctly rank order preferences between algorithms. The evaluation metric or the development set or test set should be charged.

The misclassification error can be written as a function as follows:

$$Error : \frac{1}{m_{dev}} \sum_{i=1}^{m_{dev}} \mathcal{L}\{\hat{y}^{(i)} \neq y^{(i)}\}$$

This function counts up the nubmer of misclassified examples.

The problem with this evaluation metric is that it treats pornographic vs non-pornographic images equally. On way to change this evaluation metrics is to add the weight term $w^{(i)}$.

$$w^{(i)} = \begin{cases} 1 & \text{if } x^{(i)} \text{ is non-pornographic} \\ 10 & \text{if } x^{(i)} \text{ is pornographic} \end{cases}$$

The function becomes:

$$Error : \frac{1}{\sum_w^{(i)}} \sum_{i=1}^{m_{dev}} w^{(i)} \mathcal{L}\{\hat{y}^{(i)} \neq y^{(i)}\}$$

Guideline

1. Define correctly an evaluation metric that helps better rank order classifiers
2. Optimize the evaluation metric
3. If doing well on your metric + dev/test set does not correspond to doing well on your application, change your metric and/or dev/test set.

1.3 Comparing to human-level performance

1.3.1 Why human-level performance?

Today, machine learnning algorithms can compete with human-level performance since they are more productive and more feasible in a lot of application. Also, the workflow of designing and building a machine learning system, is much more efficient than before.

Moreover, some of the tasks that humans do are close to "perfection", which is why machine learning tries to mimic human-level performance.

Figure 6 shows the performance of humans and machine learning over time.

Machine learning progresses slowly when it surpasses human-level performance. One of the reason is that human-level performance can be close to Bayes optimal error, especially for natural perception problem.

Bayes optimal error is defined as the best possible error. In other words, it means that any functions mapping from x to y can't surpass a certain level of accuracy.

Also, when the performance of machine learning is worse than the performance of humans, you can improve it with different tools. They are harder to use once it's surpasses human-level performance. These tools are:

- Get labeled data from humans
- Gain insight from manual error analysis: Why did a person get this right?
- Better analysis of bias/variance.

1.3.2 Avoidable bias

By knowing what the human-level performance is, it is possible to tell when a training set is performing well or not.

In the case that Figure 7 shows, the human-level error as a proxy for Bayes error since humans are good to identify images. If you want to improve the performance of the training set but you can't do better than the Bayes error otherwise the training set is overfitting. By knowing the Bayes error, it is easier to focus on whether bias or variance avoidance tactics will improve the performance of the model.

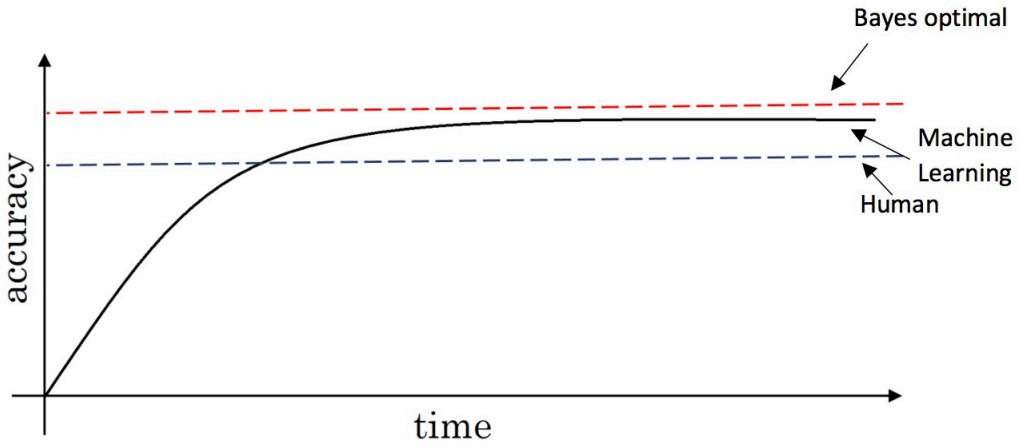

Figure 6: The performance of humans and machine learning over time.

	Classification error (%)	
	Scenario A	Scenario B
Humans	1	7.5
Training error	8	8
Development error	10	10

Figure 7: Different human-level in different scenarios

Scenario A There is a 7% gap between the performance of the training set and the human-level error. It means that the algorithm isn't fitting well with the training set since the target is around 1%. To resolve the issue, we use bias reduction technique such as training a bigger neural networkk or running the training set longer.

Scenario B The training set is doing good since there is only a 0.5% difference with the human-level error. The difference between the training set and the human-level error is called avoidable bias. The focus here is to reduce the variance since the difference between the training set error and the development error is 2%. To resolve the issue, we use variance reduction technique such as regularization or have a bigger training set.

1.3.3 Understanding human-level performance

Human-level error gives an estimate of Bayes error.

	Classification error (%)
Typical human	3.0
Typical doctor	1.0
Experienced doctor	0.7
Team of experienced doctors	0.5

Figure 8: Medical image classification

Example 1: Medical image classification This is an example of a medical image classification in which the input is a radiology image and the output is a diagnosis classfication decision.

The definition of human-level error depends on the purpose of the analysis, in this case, by definition the Bayes error is lower or equal to 0.5%.

	Classification error (%)		
	Scenario A	Scenario B	Scenario C
Human (proxy for Bayes error)	1	1	0.5
	0.7	0.7	
	0.5	0.5	
Training error	5	1	0.7
Development error	6	5	0.8

Figure 9: Error analysis in different scenarios according to human-levels

Example 2: Error analysis

Scenario A In this case, the choice of human-level performance doesn't have an impact. The avoidable bias is between 4%-4.5% and the variance is 1%. Therefore, the focus should be on bias reduction technique.

Scenario B In this case, the choice of human-level performance doesn't have an impact. The avoidable bias is between 0%-0.5% and the variance is 4%. Therefore, the focus should be on variance reduction technique.

Scenario C In this case, the estimate for Bayes error has to be 0.5% since you can't go lower than the human-level performance otherwise the training set is overfitting. Also, the avoidable bias is 0.2% and the variance is 0.1%. Therefore, the focus should be bias reduction technique.

Summary of bais/variance with human-level performance

- Human-level error — proxy of Bayes error
- If the difference between human-level error and the training error is bigger than the difference between the training error and the development error. The focus should be on bias reduction technique
- If the difference between training error and the development error is bigger than the difference between the human-level error and the training error. The focus should be on variance reduction technique.

1.3.4 Surpassing human-level performance

	Classification error (%)	
	Scenario A	Scenario B
Team of humans	0.5	0.5
One human	1.0	1
Training error	0.6	0.3
Development error	0.8	0.4

Figure 10: Classification task

Example 1: Classification task

Scenario A In this case, the Bayes error is 0.5%, therefore the available bias is 0.1% et the variance is 0.2%.

Scenario B In this case, there is not enough information to know if bias reduction or variance reduction has to be done on the algorithm. It doesn't mean that the model cannot be improve, it means that the conventional ways to know if bias reduction or variance reduction are not working in this case.

There are many problems where machine learning significantly surpass human-level performance, especially with structured data:

- Online advertising
- Product recommendations
- Logistics (predicting transmit time)
- Loan approvals

1.3.5 Improving your model performance

The two fundamental assumptions of supervised learning There are 2 fundamental assumptions of supervised learning. The first one is to have a low avoidable bias which means that the training set fits well. The second one is to have a low or acceptable variance which means that the training set performance generalizes well to the development set and test set. The strategy of improving your model can be seen in Figure 11.

If the different between human-level error and the training error is bigger than the difference between the training error and the development error, the focus should be on bias reduction technique which are training a bigger model, training longer or change the neural networks architecture or try various hyperparameters search.

If the difference between training error and the development error is bigger than the difference between the human-level error and the training error, the focus should be on variance reduction technique which are bigger dataset, regularization or change the neural networks architectures or try various hyperparameters search.

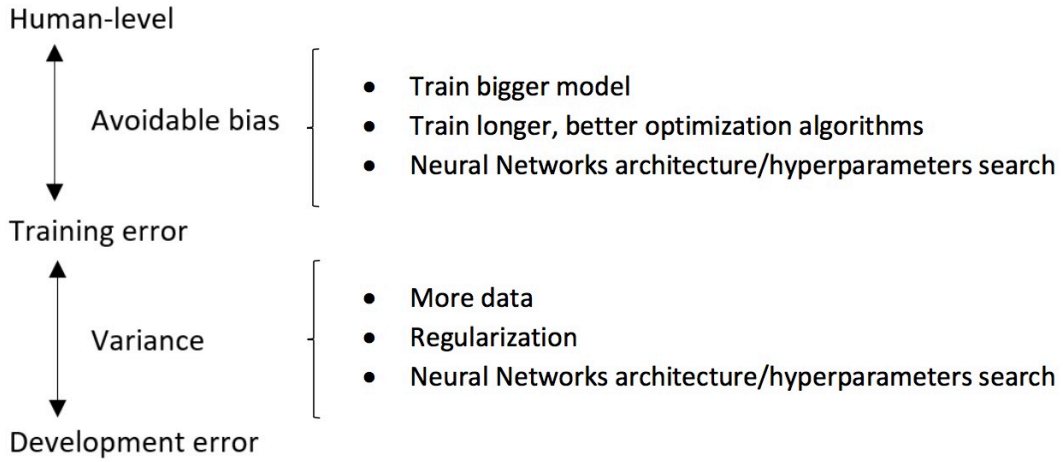

Figure 11: The strategy to improve your models.

1.4 Error Analysis

1.4.1 Carrying out error analysis

Look at dev examples to evaluate ideas Should you try to make your cat classifier do better on dogs?

Error analysis:

- Get ∼100 mislabeled dev set examples.
- Count up how many are dogs.

Evaluate multiple ideas in parallel Ideas for cat detection:

- Fix pictures of dogs being recognized as cats
- Fix great cats (lions, panthers, etc…) being misrecognized
- Improve performance on blurry images

1.4.2 Cleaning up incorrectly labeled data

DL algorithms are quite robust to random errors in the training set.

Look insight of the overall dev set errors, and compare between the errors due to incorrect labels and the errors due to other causes. The goal of dev set is to help you select between classifier A & B.

Correcting incorrect dev/test set examples

- Apply same process to your dev and test sets to make sure they continue to come from the same distribution.
- Consider examining examples your algorithm got right as well as ones it got wrong.
- Train and dev/test data may now come from slightly different distributions.

1.4.3 Build your first system quickly, then iterate

Depending on the area of application, the guideline below will help you prioritize when you build your system.

Guideline

1. Set up development/test set and metrics
    - Set up a target
2. Build an initial system quickly
    - Train training set quickly: Fit the parameters
    - Development set: Tune the parameters
    - Test set: Assess the performance
3. Use Bias/Variance analysis & Error analysis to prioritize next steps

1.5 Mismatched training and dev/test set

1.5.1 Training and testing on different distributions

Example Cat vs Non-cat In this example, we want to create a mobile application that will classify and recognize pictures of cats taken and uploaded by users.

There are two sources of data used to develop the mobile app. The first data distribution is small, 10,000 pictures uploaded from the mobile application. Since they are from amateur users, the pictures are not professionally shot, not well framed and blurrier. The second source is from the web, you downloaded 200,000 pictures where cat's pictures are professionally framed and in high resolution.

The problem is that you have a different distribution:

1. small data set from pictures uploaded by users. This distribution is important for the mobile app.
2. bigger data set from the web.

The guideline used is that you have to choose a development set and test set to reflect data you expect to get in the future and consider important to do well.

The data is split as the Figure 12. The advantage of this way of splitting up is that the target is well defined.

The disadvantage is that the training distribution is different from the development and test set distributions. However, this way of splitting the data has a better performance in long term.

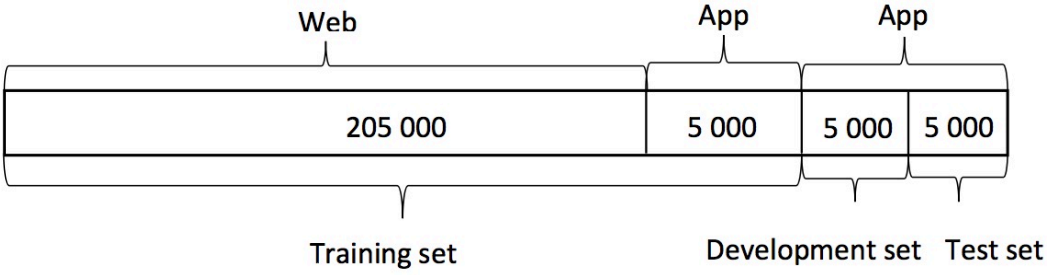

Figure 12: The split of the data, which have

1.5.2 Bias and Variance with mismatched data distributions

Example: Cat classifier with mismatched data distribution When the training set is from a different distribution than the development and test sets, the method to analyze bias and variance changes, see Figure 13.

	Classification error (%)					
	Scenario A	Scenario B	Scenario C	Scenario D	Scenario E	Scenario F
Human (proxy for Bayes error)	0	0	0	0	0	4
Training error	1	1	1	10	10	7
Training-development error	-	9	1.5	11	11	10
Development error	10	10	10	12	20	6
Test error	-	-	-	-	-	6

Figure 13: Bias and variance with mismatched data.

Scenario A If the development data comes from the same distribution as the training set, then there is a large variance problem and the algorithm is not generalizing well from the training set.

However, since the training data and the development data come from a different distribution, this conclusion cannot be drawn. There isn't necessarily a variance problem. The problem might be that the development set contains images that are more difficult to classify accurately.

When the training set, development and test sets distributions are different, two things change at the same time. First of all, the algorithm trained in the training set but not in the development set. Second of all, the distribution of data in the development set is different. It's difficult to know which of these two changes what produces this 9% increase in error between the training set and the development set. To resolve this issue, we define a new subset called training-development set. This new subset has the same distribution as the training set, but it is not used for training the neural network.

Scenario B The error between the training set and the training-development set is 8%. In this case, since the training set and training-development set come from the same distribution, the only difference between them is the neural network sorted the data in the training and not in the training development. The neural network is not generalizing well to data from the same distribution that it hadn't seen before. Therefore, we have really a variance problem.

Scenario C In this case, we have a mismatch data problem since the 2 data sets come from different distribution.

Scenario D In this case, the avoidable bias is high since the difference between Bayes error and training error is 10%.

Scenario E In this case, there are 2 problems. The first one is that the avoidable bias is high since the difference between Bayes error and training error is 10% and the second one is a data mismatched problem.

Scenario F Development should never be done on the test set. However, the difference between the development set and the test set gives the degree of overfitting to the development set.

Figure 14 shows the general formulation.

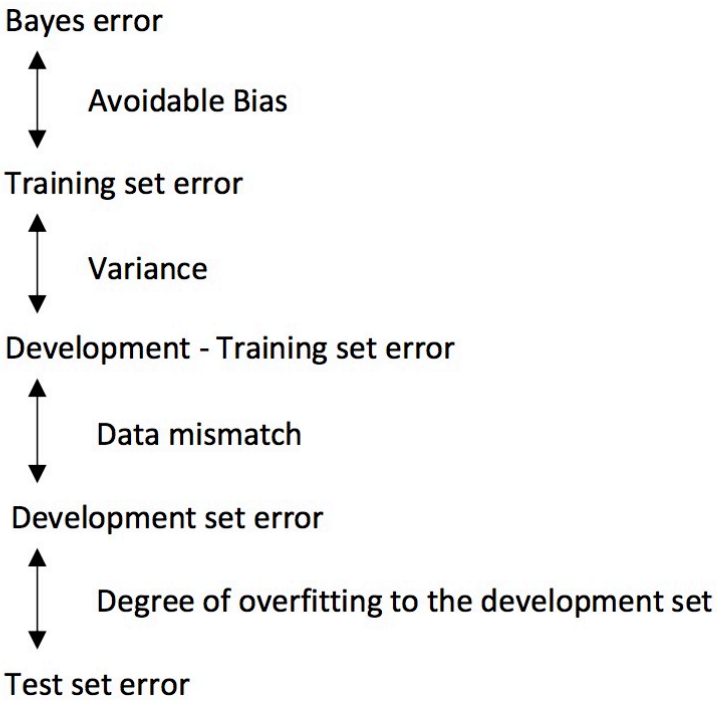

Figure 14: The general formulation of bias and variance with mismatched data.

1.5.3 Addressing data mismatch

This is a general guideline to address data mismatch:

- Perform manual error analysis to understand the error differences between training, development/test sets. Development should never be down on test set to avoid overfitting.
- Make training data or collect data similar to development and test sets. To make the training data more similar to your development set, you can use is artificial data synthesis. However, it is possible that if you might be accidentally simulating data only from a tiny subset of the space all possible examples.

1.6 Learning from multiple tasks

1.6.1 Transfer learning

Transfer learning refers to using the neural network knowledge from for another application.
When to use transfer learning

- Task A and B have same input x
- A lot more data for Task A than Task B
- Low level features from Task A could be helpful for Task B

Example 1: Cat recognition — radiology diagnosis The following neural network is trained for cat recognition, but we want to adapt it for radiology diagnosis. The neural network will learn about the structure and the nature of images. This initial phase of training on image recognition is called pre-training, since it will pre-initialize the weights of the neural network. Updating all the weights afterwards is called fine-tuning.

For cat recognition

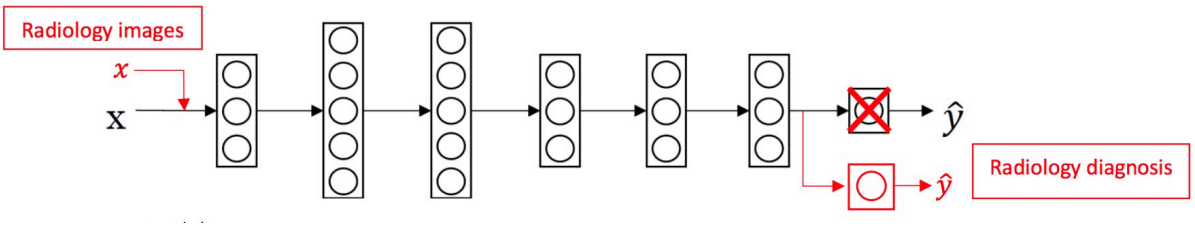

Figure 15: Transfer learning example, from Task cats recognition to Task radiology diagnosis.

- Input x: image
- Output y: 1—cat; 0—non-cat

For Radiology diagnosis

- Input x: Radiology images — CT Scan, X-rays
- Output y: Radiology diagnosis, 1—tumor malign; 0—tumor benign

Guideline

- Delete last layer of neural network
- Delete weights feeding into the last output layer of the neural network
- Create a new set of randomly initialized weights for the last layer only
- New dataset (x, y)

1.6.2 Multi-task learning

Multi-task learning refers to having one neural network do simultaneously several tasks.
When to use multi-task learning

- Training on a set of tasks that could benefit from having shared lower-level features
- Usually: Amount of data you have for each task is quite similar
- Can train a big enough neural network to do well on all tasks

Example: Simplified autonomous vehicle The vehicle has to detect simultaneously several things: pedestrians, cars, road signs, traffic lights, cyclists, etc. We could have trained four separate neural networks, instead of train one to do four tasks. However, in this case, the performance of the system is better when one neural network is trained to do four tasks than training four separate neural networks since some of the earlier features in the neural network could be shared between the different types of objects.

The input $x^{(i)}$ is the image with multiple labels. The output $y^{(i)}$ has 4 labels which are represents:

$$\boldsymbol{y^{(i)}} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \begin{matrix} \text{Pedestrians} \\ \text{Cars} \\ \text{Road signs - Stop} \\ \text{Traffic lights} \end{matrix} \longrightarrow \quad \boldsymbol{Y} = \begin{bmatrix} | & | & | & | \\ \boldsymbol{y^{(1)}} & \boldsymbol{y^{(2)}} & \boldsymbol{y^{(3)}} & \boldsymbol{y^{(4)}} \\ | & | & | & | \end{bmatrix} \qquad \boldsymbol{Y} \in \mathbb{R}^{(4,m)}$$

To train this neural network, the cost function is defined as follow:

$$-\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{4}(y_j^{(i)}\log(\hat{y}_j^{(i)})) + (1-y_j^{(i)})\log(1-\hat{y}_j^{(i)}))$$

Also, the cost can be compute such as it is not influenced by the fact that some entries are not labeled. For example:

$$\boldsymbol{Y} = \begin{bmatrix} 1 & 0 & ? & ? \\ 0 & 1 & ? & 0 \\ 0 & 1 & ? & 1 \\ ? & 0 & 1 & 0 \end{bmatrix}$$

1.7 End-to-end deep learning

1.7.1 What is end-to-end deep learning?

End-to-end deep learning is the simplification of a processing or learning systems into one neural network.

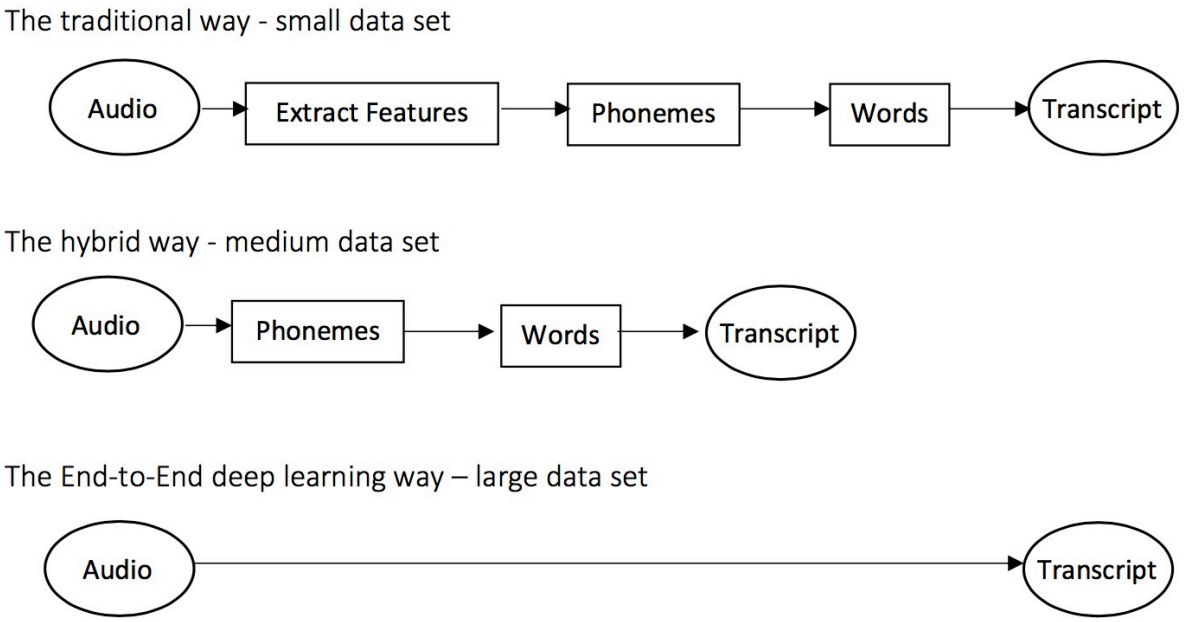

Figure 16: Different ways to train a speech recognition model.

Example: Speech recognition model End-to-end deep learning cannot be used for every problem since it needs a lot of labeled data. It is used mainly in audio transcripts, image captures, image synthesis, machine translation, steering in self-driving cars, etc.

1.7.2 Whether to use end-to-end deep learning

Before applying end-to-end deep learning, you need to ask yourself the following question: Do you have enough data to learn a function of the complexity needed to map x and y?

Pros:

- Let the data speak
  - By having a pure machine learning approach, the neural network will learn from x to y. It will be able to find which statistics are in the data, rather than being forced to reflect human preconceptions.
- Less hand-designing of components needed
  - It simplifies the design work flow.

Cons:

- Large amount of labeled data
  - It cannot be used for every problem as it needs a lot of labeled data.
- Excludes potentially useful hand-designed component
  - Data and any hand-design's components or features are the 2 main sources of knowledge for a learning algorithm. If the dataset is small, a hand-design system is a way to give manual knowledge into the algorithm.